International Journal of Electronics and Communication Engineering and Technology (IJECET)
Volume 12, Issue 3, September–December 2021, pp. 11-18. Article ID: IJECET_12_03_002
Available online at https://iaeme.com/Home/issue/IJECET?Volume=12&Issue=3
ISSN Print: 0976-6464 and ISSN Online: 0976-6472
DOI: https://doi.org/10.34218/IJECET.12.3.2021.002

ACCELERATED DEEP LEARNING INFERENCE FROM CONSTRAINED EMBEDDED DEVICES

Bhargav Bhat[1] and **Abhay A Deshpande**[1]
[1]Department of Electronics and Communication Engineering,
RV College of Engineering (RVCE), Bengaluru, Karnataka, India

ABSTRACT

Hardware looping is a feature of some processor instruction sets whose hardware can repeat the body of a loop automatically, rather than requiring software instructions which take up cycles (and therefore time) to do so. Loop Unrolling is a loop transformation technique that attempts to advance a program's execution speed to the detriment of its twofold size, which is a methodology known as space–time tradeoff. A convolutional neural network is created with simple loops, with hardware looping, with loop unrolling and with both hardware looping and loop unrolling, and a comparison is made to evaluate the effectiveness of hardware looping and loop unrolling. The hardware loops alone will add to a cycle check decline, while the mix of hardware loops and dot product instructions will decrease the clock cycle tally further. The CNN is simulated on Xilinx Vivado 2021.1 running on Zync-7000 FPGA.

Keywords: Convolutional Neural Network, Deep Learning, FPGA, Hardware Looping, Loop Unrolling, Vivado.

Cite this Article: Bhargav Bhat and Abhay A Deshpande, Accelerated Deep Learning Inference from Constrained Embedded Devices, *International Journal of Electronics and Communication Engineering and Technology*, 12(3), 2021, pp. 11-18.
https://iaeme.com/Home/issue/IJECET?Volume=12&Issue=3

1. INTRODUCTION

Profound learning calculations have seen achievement in a wide assortment of uses, for example, machine interpretation, picture and discourse acknowledgment, and self-driving vehicles. In any case, these calculations have as of late acquired traction in the installed frameworks space. Most implanted frameworks depend on modest microcontrollers with restricted memory limit, and, subsequently, are commonly seen as not equipped for running profound learning calculations. Nevertheless, we think about that progressions in pressure of neural organizations and neural organization engineering, combined with an improved guidance set design, could make microcontroller-grade processors appropriate for explicit low-force profound learning applications. Such intricacy is substantially a lot for memory obliged microcontrollers that have memory sizes indicated in kilobytes. Some embedded system designer's work around the quandary of restricted resources by processing neural networks in the cloud. Nonetheless, this arrangement is restricted to regions with access to the Web. Cloud

preparing likewise has different burdens, for example, protection concerns, security, high idleness, correspondence power utilization, and dependability. These calculations perform massive arithmetic computations. To accelerate these calculations at a sensible cost in equipment, we can utilize an instruction set extension comprised of two instruction types-hardware loops and dot product instructions. The primary commitments of this paper are as per the following:

- We propose an approach for computing neural network functions that are advanced for the utilization of hardware loops and dot product instructions.
- The effectiveness of hardware loops and dot product instructions for performing deep learning functions are evaluated, and
- To perform Lenet-5 Neural Network on Zync-7000

There have been different ways to deal with accelerate profound learning capacities. The methodologies can be sorted into two sets. In the first set, the approaches which which attempt to enhance the size of the neural organizations, or, all in all, upgrade the software. Approaches in the second set try to optimize the equipment or hardware on which neural networks are running. As the set used here deals with hardware optimization, we will focus on the related approaches for hardware optimization, and only momentarily observe the progress in software optimizations. A simple instruction set is proposed to evaluate the effectiveness of hardware loops and dot product instructions with fully upgraded assembly functions for the fully connected convolutional neural network [1]. A custom instruction set architecture is used for efficient realization of artificial neural networks and can be parameterized to a subjective fixed-point design [2]. A CNN –specific Instruction set architecture is used which deploys the instruction parameters with high flexibility which embeds parallel computation and data reuse parameters in the instructions [3]. The instruction extensions and micro architectural advancements to increase computational thickness and to limit the amount of pressure towards shared memory hierarchy in RISC Processors [4]. A repetitive neural organization into a convolutional neural network, and the profound provisions of the picture were learnt in equal utilizing the convolutional neural network and the intermittent neural organization [5]. The framework of Complex Network Classifier (CNC) is used by integrating network embedding and convolutional neural network to tackle the problem of network classification. By training the classifier on synthetic complex network data, they showed that CNC can not only classify networks with high accuracy and robustness but can also extract the features of the networks automatically [6]. A valued prediction method is used to exploit the spatial correlation of zero-valued activations within the CNN output feature maps, thereby saving convolution operations [7]. The impact of packet loss on data integrity is reduced by taking advantage of the deep network's ability to understand neural data and by using a data repair method based on convolutions neural network [8]. The instruction set simulation process is used soft-core Reduced Instruction Set Computer (RISC) processor. They provided reliable simulation platform in creating customizable instruction set for Application Specific Instruction Set Processor (ASIP) [9]. RISC-V ISA compatible processor and effects of instruction set is analyzed on the pipeline/micro-architecture design in terms of instructions encoding, functionality of instructions, instruction types, decoder logic complexity, data hazard detection, register file organization and access, functioning of pipeline, effect of branch instructions, control flow, data memory access, operating modes and execution unit hardware resources [10]. Deep learning algorithms are used increasingly in smart applications. Some of them also run in Internet of Things (IoT) devices. IoT Analytics reports that, by 2025, the number of IoT devices will rise to 22 billion. The motivation for our work stems from the fact that the rise of the IoT will increase the need for low-cost devices built around a single microcontroller capable of

supporting deep learning algorithms. Accelerating deep learning inference in constrained embedded devices, presented in this paper, is our attempt in this direction.

The remainder of this paper is organized as follows. Section II presents the related work in hardware and software enhancements aimed at accelerating the neural network computation. Section III shows the methodology concerned with the project. Section IV shows the results and the subsequent discussions regarding the obtained results. Section V presents the conclusion and plans for additional work for the future.

2. BACKGROUND

2.1. Convolutional Neural Networks

A convolutional neural organization (CNN, or ConvNet) is a class of artificial neural organization, most generally applied to understand visual imagery. A CNN is a deep learning algorithm which can can take in an information picture, allot significance, to different viewpoints/objects in the picture and have the option to separate or differentiate one from the other. The pre-processing needed in a CNN is a lot of lower when contrasted with other characterization calculations. While in crude techniques channels are hand-designed, with enough preparing, CNNs can gain proficiency with these channels/qualities. CNN has a lot of applications in decoding facial recognition, analyzing documents, historic and environmental collections, understanding climate, grey areas to see holistic view of what a human sees, advertising and other fields.

2.2. Lenet

Lenet (also called Lenet-5) is an exemplary convolutional neural network which utilizes convolutions, pooling and fully connected layers. Lenet is used for handwritten digit recognition with the MNIST dataset.

2.3. MNSIT Dataset

MNIST is the acronym for Modified National Institute of Standards and Technology database. The MNIST database contains 60,000 training images and 10,000 testing images. MNIST dataset pictures have the measurements 28 x 28. To get the MNIST pictures measurement to the meet the necessities of the input layer, the 28 x 28 pictures are cushioned or padded. Some of the test pictures from MNIST test dataset are as shown in Fig 1.

Figure 1 Some Test pictures from MNIST dataset

2.4. Building Blocks of Convolutional Neural Network

- Convolutional Layer is the core center structure of CNN. In a CNN, the information is a tensor with a shape: (number of data sources) x (input height) x (input width) x (input channels). In the wake of going through a convolutional layer, the picture becomes disconnected to an element map, likewise called an actuation map, with shape: (number of data sources) x (highlight map height)) x (include map width) x (feature map channels). Convolutional layers convolve the information and pass its outcome to the following layer.
- Pooling Layer: Convolutional networks may incorporate local and/or global pooling layers alongside traditional convolutional layers. Pooling layers diminish the components of information by combining the outputs of neuron clusters at one layer into a single neuron in the following layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map. There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map, while average pooling takes the average value.
- Fully Connected Layers: Fully connected layers connect every neuron in one layer to each neuron in another layer. It is equivalent to a conventional multi-layer perceptron neural network (MLP). The smoothed network goes through a fully connected layer to characterize the pictures.
- Receptive field: In neural networks, each neuron gets input from some number of locations in the past layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's receptive field. Ordinarily the area is a square (e.g. 5 by 5 neurons). Whereas, in a fully connected layer, the receptive field is the entire past layer. Accordingly, in each convolutional layer, each neuron takes input from a bigger region in the input than previous layers. This is expected due to applying the convolution over and over, which takes into account the value of a pixel, as well as its encompassing pixels. When utilizing dilated layers, the number of pixels in the receptive field remains constant, but the field is more scantily populated as its measurements grow when combining the impact of several layers.
- Weights: Every neuron in a neural network registers an output value by applying a particular function to the input values received from the receptive field in the past layer. The function that is applied to the input values is dictated by a vector of weights and a bias (ordinarily real numbers). Learning consists of iteratively adjusting these biases and weights.
- The vector of weights and the bias are called filters and represent specific features of the input (e.g., a specific shape). A distinctive feature of CNNs is that many neurons can share the same filter. This decreases the memory footprint because a single bias and a single vector of weights are utilized across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.

3. LENET ARCHITECTURE

Lenet is a pre-trained Convolutional Neural Network Model used for recognizing handwritten and machine-printed characters. The organization has 5 layers with learnable boundaries and thus named Lenet-5. It has three arrangements of convolution layers with a blend of average pooling. After the convolution and average pooling layers, we have two fully connected layers. Finally, a Softmax classifier is used to characterize the pictures into separate class. The lenet layers are depicted in the following fig 2.

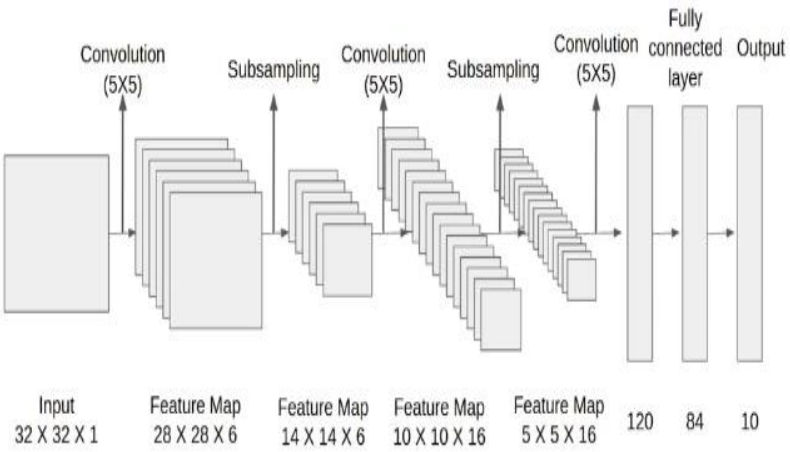

Figure 2 Architecture of Lenet-5 Model

The following table is used to understand the architecture in more detail.

Table 1

Layer	#filters/ Neuron	Filter Size	Stride	Size of Feature map	Activation Function
Input	-	-	-	32x32x1	
Conv 1	6	5 x 5	1	28x28x6	tanh
Avg Pooling 1		2 x 2	2	14x14x6	
Conv 2	16	5 x 5	1	10x10x16	tanh
Avg Pooling 2		2 x 2	2	5x5x16	
Conv 3	120	5 x 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

The first layer is the input layer with highlight map size 32 x 32 x 1.

Then, at that point, we have the first convolution layer with 6 channels of size 5 x 5 and step is 1. The initiation work utilized at his layer is hyperbolic tangent (tanh). The output feature map is 28 x 28 x 6.

Then, we have an average pooling layer with channel size 2 x 2 and step 1. The subsequent component map is 14 x 14 x 6. Since the pooling layer doesn't influence the quantity of channels.

After this comes the second convolution layer with 16 filters of 5 x 5 and step 1. Also, the activation function is tanh. Now the output size is 10 x 10 x 16.

Again comes the other average pooling layer of 2 x 2 with step 2. As a result, the size of the feature map reduced to 5 x5 x16.

The final pooling layer has 120 filters of 5 x5 with stride 1 and activation function tanh. Now the output size is 120.

The next is a fully connected layer with 84 neurons that result in the output to 84 values and the activation function used here is again tanh.

The last layer is the output layer with 10 neurons and Softmax function. The Softmax gives the likelihood that an information point has a place with a specific class. The most elevated worth is then anticipated.

This is the entire architecture of the Lenet-5 model. The number of trainable parameters of this architecture is around 60,000.

4. RESULTS

The following table gives the hardware cost occupied by our design on the Zync-7000 board. It shows that the design occupies 47% of LUTs, 19% of LUTRAMs, 28% of FFs, 59% of BRAMs, 54% of DSP, and 3% of BUFG. Although, the implemented design can be displayed to gives us idea on how the design have been distributed, placed and routed on the selected Zync-7000 board.

Table 2

Resource	Utilization	Available	Utilization %
LUT	25456	53200	47.85
LUTRAM	3478	17400	19.99
FF	30456	106400	28.62
BRAM	83.5	140	59.64
DSP	120	220	54.55
BUFG	1	32	3.13

4.1. Software Model (MATLAB)

In the software model goes through the whole LeNet CNN and give us its prediction. Fig 3 depicts result of the software model on matlab.

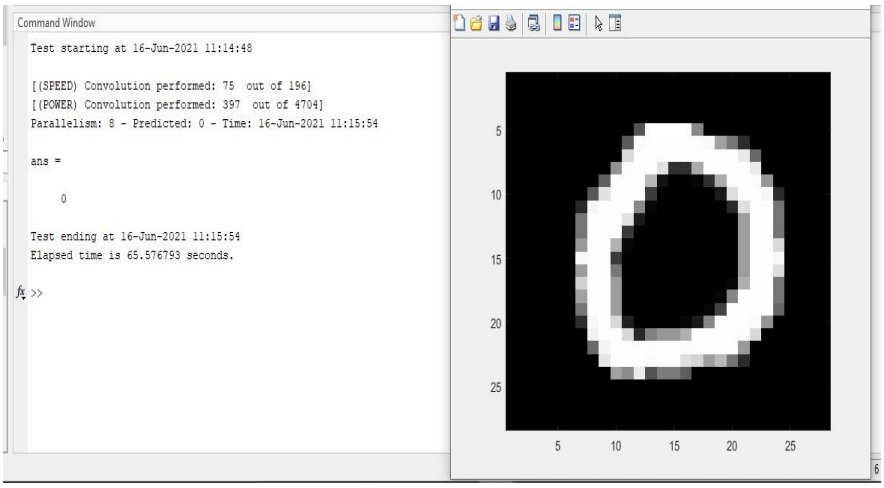

Figure 3 Matlab Result

4.2. Hardware Model

For the Hardware model, it gives us 10 outputs for the score of all 10 digits which are digit 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The highest score should be the correct answer. So, you can see from all images the the highest score is the correct value. Fig 4 shows the simulation result from Xilinx Vivado. In the below example where the number 0 has been taken, the highest score 5 shows that 0 is the predicted number.

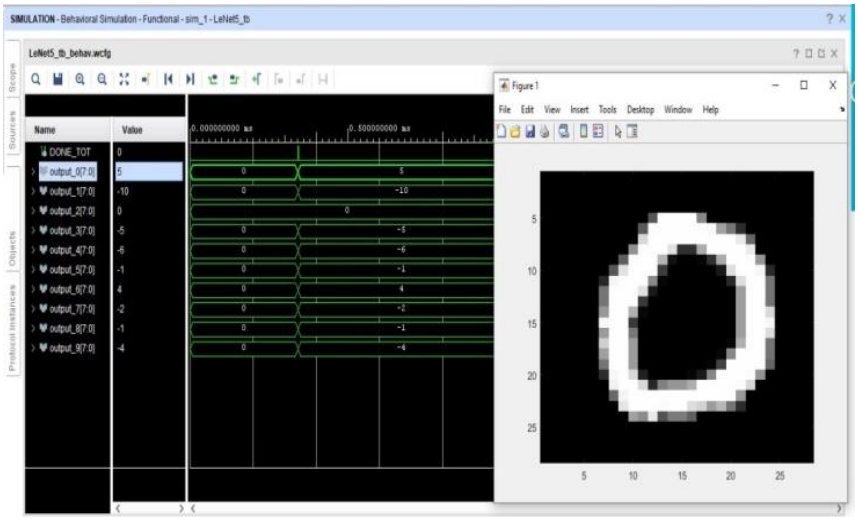

Figure 4 Behavioral simulation on Xilinx Vivado

4.3. Implementation

Once the implementation is reached the implementation summary that summarizes all implantation report will be provided. The Fig 5 depicts the implemented design.

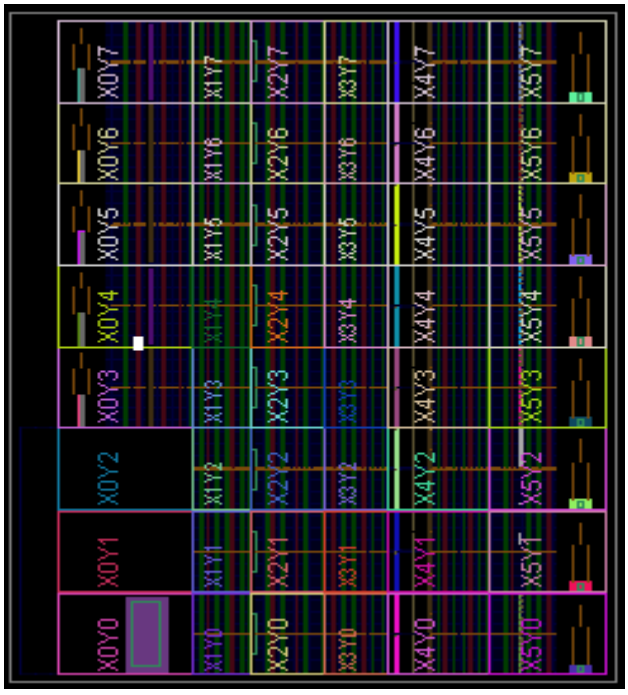

Figure 5 Implemented Design of Lenet-5

5. CONCLUSION

In this paper, we proposed Lenet-5 Convolutional Neural Network with optimizations done using hardware looping and dot product units, which provided high accuracy when recognizing handwritten data using the MNSIT dataset. The hardware loops alone contribute to 24% cycle count decrease, while the dot products reduce the cycle count by 27%. As embedded systems are highly price-sensitive, this is an important consideration. Getting the sizes of neural networks down is an essential step in expanding the possibilities for neural networks in embedded systems.

An interesting topic for further research is Posit - an alternative floating-point number format that may offer additional advantages, as it has an increased dynamic range at the same word size. Because of the improved dynamic range, weights could be stored in lower precision, thus, again, decreasing the memory requirements. Combining the reduced size requirements with low-cost ISA improvements could make neural networks more ubiquitous in the price-sensitive embedded systems market.

REFERENCES

[1] J. Vreca et al., "Accelerating Deep Learning Inference in Constrained Embedded Devices Using Hardware Loops and a Dot Product Unit," in *IEEE Access*, vol. 8, pp. 165913-165926, 2020, doi: 10.1109/ACCESS.2020.3022824.

[2] D. Valencia, S. F. Fard and A. Ali mohammad, "An Artificial Neural Network Processor with a Custom Instruction Set Architecture for Embedded Applications," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5200-5210, Dec. 2020, doi: 10.1109/TCSI.2020.3003769.

[3] X. Chen and Z. Yu, "A Flexible and Energy-Efficient Convolutional Neural Network Acceleration with Dedicated ISA and Accelerator," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 7, pp. 1408-1412, July 2018, doi: 10.1109/TVLSI.2018.2810831.

[4] M. Gautschi, Pasquale Davide Schiavone, Andreas Traber, Igor Loi, "Near-Threshold RISC-V Core with DSP Extensions for Scalable IoT Endpoint Devices," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700-2713, Oct. 2017, doi: 10.1109/TVLSI.2017.2654506.

[5] Y. Tian, "Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm," in *IEEE Access*, vol. 8, pp. 125731-125744, 2020, doi: 10.1109/ACCESS.2020.3006097.

[6] Xin, J. Zhang and Y. Shao, "Complex network classification with convolutional neural network," in Tsinghua Science and Technology, vol. 25, no. 4, pp. 447-457, Aug. 2020, doi: 10.26599/TST.2019.9010055.

[7] Shomron and U. Weiser, "Spatial Correlation and Value Prediction in Convolutional Neural Networks," in *IEEE Computer Architecture Letters*, vol. 18, no. 1, pp. 10-13, 1 Jan.-June 2019, doi: 10.1109/LCA.2018.2890236

[8] Y. Qie, P. Song and C. Hao, "Data Repair Without Prior Knowledge Using Deep Convolutional Neural Networks," in *IEEE Access*, vol. 8, pp. 105351-105361, 2020, doi: 10.1109/ACCESS.2020.2999960.

[9] A. J. Salim, S. I. M. Salim, N. R. Samsudin and Y. Soo, "Customized instruction set simulation for soft-core RISC processor," 2012 *IEEE Control and System Graduate Research Colloquium, Shah Alam, Selangor*, 2012, pp. 38-42, doi: 10.1109/ICSGRC.2012.6287132.

[10] A. Raveendran, V. B. Patil, D. Selvakumar and V. Desalphine, "A RISC-V instruction set processor-micro-architecture design and analysis," 2016 *International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA)*, Bangalore, 2016, pp. 1-7, doi: 10.1109/VLSI-SATA.2016.7593047.